Variable ordering for taylor expansion diagrams

Daniel Gomez-Prado, Quian Ren, Serkan Askar, Maciej Ciesielski, Emmanuel Boutillon

VARIABLE ORDERING FOR TAYLOR EXPANSION DIAGRAMS

Daniel Gomez-Prado[1] *Qian Ren*[1] *Serkan Askar*[1]
dgomezpr@ecs.umass.edu *qren@ecs.umass.edu* *saskar@ecs.umass.edu*

Maciej Ciesielski[1] *Emmanuel Boutillon*[2]
ciesiel@ecs.umass.edu *emmanuel.boutillon@univ.ubs.fr*

[1]*Department of Electrical and Computer Engineering, University of Massachusetts, Amherst, USA*
[2]*Lester, Université de Bretagne Sud, Lorient, France*

Abstract: *This paper presents an algorithm for variable ordering for Taylor Expansion Diagrams (TEDs). First we prove that the function implemented by the TED is independent of the order of its variables, and then that swapping of two adjacent variables in a TED is a local permutation similar to that in BDD. These two properties allow us to construct an algorithm to swap variables locally without affecting the entire TED. The proposed algorithm can be used to perform dynamic reordering, such as sifting or window permutation. We also propose a static ordering that can help reduce the permutation space and speed up the search of an optimal variable order for TEDs.*

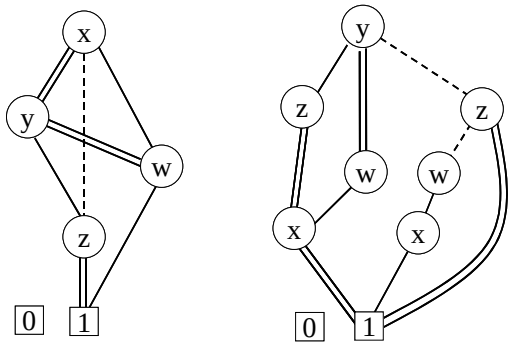
Introduction

It has been demonstrated that TEDs are a compact, canonical, graph-based representation for algebraic expressions and boolean functions, subject to the imposition of a total ordering on the variables [1]. This diagram, due to its compactness and canonicity property, can be exploited to facilitate equivalence checking of high level representations of digital designs containing arithmetic data-paths interspersed with random boolean logic.

The TED is obtained by using the Taylor Expansion of the function, one variable at a time. The zero derivative *f(x=0)* is a 0-child, *f '(x=0)* is a 1-child, *f "(x=0)* is a 2-child, with corresponding edges *0-edge* (dotted), *1-edge* (solid), *2-edge* (double), etc. For example figure 1 shows the construction of $f(x,w,y,z) = x^2yz^2 + x^2y^2w + z^2 + xw$ as a TED, with two different variable orders: in case a) only 4 nodes are needed to represent the TED, while in case b) we need 7 nodes to build the same TED. Thus, the size of the TED depends on the adopted variable order.

This means that the complexity of any future manipulation on a TED depends on the variable order for which it was constructed. It is therefore desirable to find the best order that minimizes the size of the TED.

The research reported in this paper has been supported in part by the National Science Foundation, contract No. CCR-0204146.



a) $x < y < w < z$ b) $y < z < w < x$

Fig. 1. Different variable ordering for $f(x,w,y,z)$.

Variable Ordering for TEDs

The TED around the origin for an univariate polynomial is the Taylor expansion

$$f(x_1) = \sum_{k=0}^{m_1} \tfrac{1}{k!} x_1^k f_{x_1}^k(0)$$

where $f(x_1) \in C^{m_1}$. In the case of multivariate polynomials the TED will be:

$$f(x_1, x_2, \ldots x_n) = \sum_{k=0}^{m_1} \tfrac{1}{k!} x_1^k \sum_{j=0}^{m_2} \tfrac{1}{j!} x_2^j \ldots \sum_{r=0}^{m_n} \tfrac{1}{r!} x_n^r f_{x_n,\ldots,x_2,x_1}^{r+\ldots+j+k}(0,0,\ldots,0)$$

Theorem I: The swap of two variables in a TED graph is a local permutation that does not affect the TED subgraph above or below the two swapped variables.

Proof: Given a multivariate polynomial $f(x_1, x_2, \ldots x_n)$, assume that we have already constructed the TED up to the variable x_{w-1} (see Fig. 2). Due to the recursive construction of the TED, this means that we have already expanded the x_{w-1} first terms of the summation.

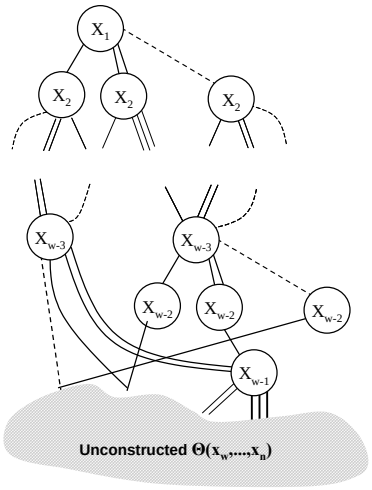


Fig. 2. Partial construction of a TED with ordering $x_1 < x_2 < \ldots < x_{w-1} < \ldots < x_n$

Here we can see that the constructed part of the TED, the top subgraph with variables above the level x_w, does not depend on the construction of the bottom subgraph, with variables below the level x_{w-1}. This is trivial to prove, as the constructed TED corresponds to the expanded and calculated parts of the summation, and the remaining construction only depends on $x_w, \ldots, x_n$. Therefore the constructed top subgraph with variables $x_1, \ldots, x_{w-1}$ remains the same for any ordering of the bottom $x_w, \ldots, x_n$ variables.

Now if we look at the unconstructed subgraph we have:

$$\Theta(x_w..x_n) = \sum_{p=0}^{m_w} \tfrac{1}{p!} x_w^p \sum_{h=0}^{m_{w+1}} \tfrac{1}{h!} x_{w+1}^h .. \sum_{r=0}^{m_n} \tfrac{1}{r!} x_n^r \frac{\partial^{r+..+h+p}}{\partial x_n^r \cdots \partial x_w^p} \Psi_{(0..0,x_w..x_n)}$$

where $\Psi_{(0,..,0,x_w,..,x_n)}$ denotes the constructed TED. It remains to be proven that the swapping of the variables x_w and x_{w+1} will not affect the construction of the subgraph $x_{w+2}, \ldots, x_n$. For polynomial, partial derivatives must be equal regardless of the order in which the differentiation is done [2]:

$$\frac{\partial^{h+p}}{\partial x_{w+1}^h \partial x_w^p} \Psi = \frac{\partial^{p+h}}{\partial x_w^p \partial x_{w+1}^h} \Psi = \Delta$$

so we can rewrite Θ as:

$$\Theta = \sum_{p=0}^{m_w} \tfrac{1}{p!} x_w^p \sum_{h=0}^{m_{w+1}} \tfrac{1}{h!} x_{w+1}^h .. \sum_{r=0}^{m_n} \tfrac{1}{r!} x_n^r \frac{\partial^{r+..+s}}{\partial x_n^r \cdots \partial x_{w+2}^s} \frac{\partial^{h+p}}{\partial x_{w+1}^h \partial x_w^p} \Psi_{(0..0,x_w..x_n)}$$

$$= \sum_{h=0}^{m_{w+1}} \tfrac{1}{h!} x_{w+1}^h \sum_{p=0}^{m_w} \tfrac{1}{p!} x_w^p .. \sum_{r=0}^{m_n} \tfrac{1}{r!} x_n^r \frac{\partial^{r+..+s}}{\partial x_n^r \cdots \partial x_{w+2}^s} \frac{\partial^{p+h}}{\partial x_w^p \partial x_{w+1}^h} \Psi_{(0..0,x_w..x_n)}$$

here we can see that the construction of the bottom subgraph

$$\Gamma_{(x_{w+2},\ldots,x_n)} = \sum_{s=0}^{m_{w+2}} \tfrac{1}{s!} x_{w+2}^s .. \sum_{r=0}^{m_n} \tfrac{1}{r!} x_n^r \frac{\partial^{r+..+s}}{\partial x_n^r \cdots \partial x_{w+2}^s} \Delta$$

remains the same under the swapping of x_w and x_{w+1}. So we have proven that swapping any variable below x_w doesn't affect Ψ and swapping any variable above x_{w+2} doesn't affect Γ; therefore swapping two adjacent variables x_w and x_{w+1} is a local permutation of the TED that doesn't change the TED subgraphs above or below the swapping variables.

A. *The algorithm for Local Swapping in TEDs*

The input of the algorithm is a TED and two adjacent variables x_w and x_{w+1} to be swapped. (see Fig. 3.)

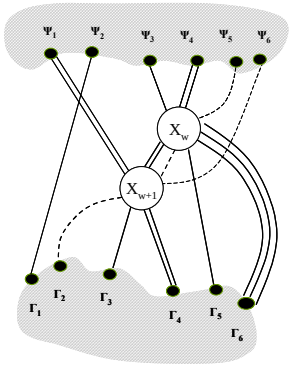


Fig. 3. TED graph with order $x_1 < \ldots < x_w < x_{w+1} < \ldots < x_n$

Step 1: Separate all x_{w+1} nodes whose parent k_p-edges have connection to x_w; split those nodes in two nodes, one with parent k_p-edges connected to x_w and the other with the remaining edges connected to the subgraph Ψ above. (see Fig. 4.a)

Step 2: Split each node x_{w+1} with parent k_p-edges connected to x_w for each child k_c-edge connected to the subgraph Γ below. (see Fig. 4.b)

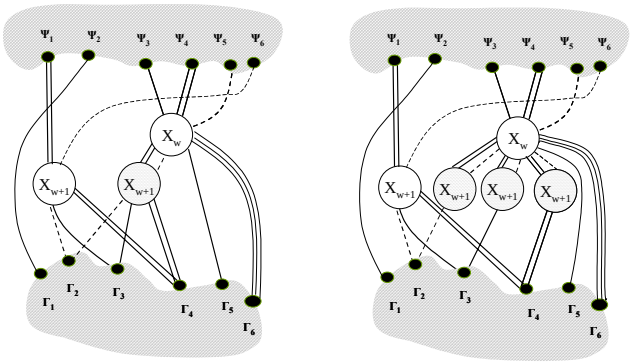


Fig. 4. Steps 1 and 2 of Local Swapping algorithm

Step 3: Separate all x_w nodes whose children k_C-edges have connection to x_{w+1}; split those nodes in two nodes, one with children k_C-edges connected to x_{w+1} and the other with the remaining edges connected to the subgraph Γ below. (see Fig. 5.a)

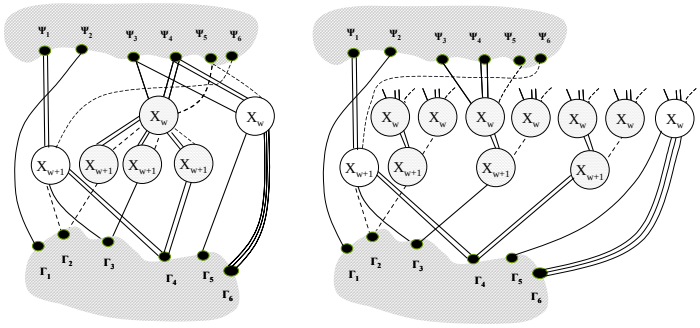


Fig. 5. Steps 3 and 4 of Local Swapping algorithm

Step 4: Split the nodes x_w with children k_C-edges connected to x_{w+1} for each child k_C-edge. (see Fig. 5.b)

Step 5: Split the nodes x_{w+1} with parent k_P-edges connected to x_w. (see Fig. 6)

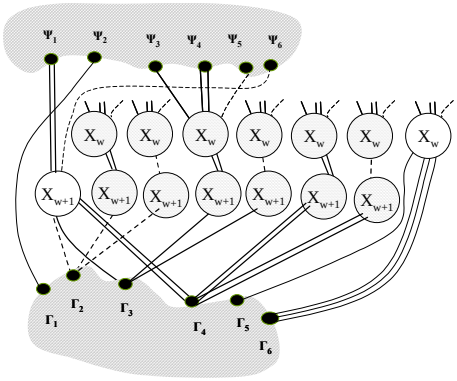


Fig. 6. Step 5

Step 6: Eliminate all redundant nodes, and swap the variables x_w and x_{w+1}. We can do this because after the fifth step we have all monomials with terms $x_w^i x_{w+1}^j$ in different paths. (see Fig. 7)

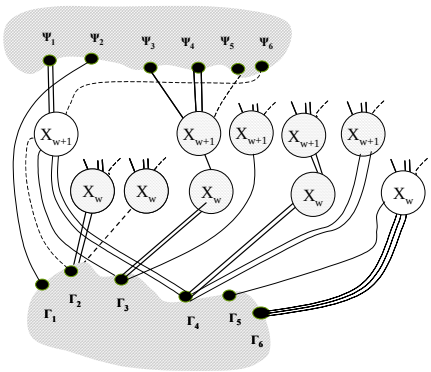


Fig. 7. Step 6

Step 7: Reestablish canonicity by means of addition. We do this by treating all different monomials as different TEDs and adding them together. This last operation is not computational expensive because all those TEDs share the same subgraph above and below, so memory doesn't need to be duplicated. (see Fig. 8)

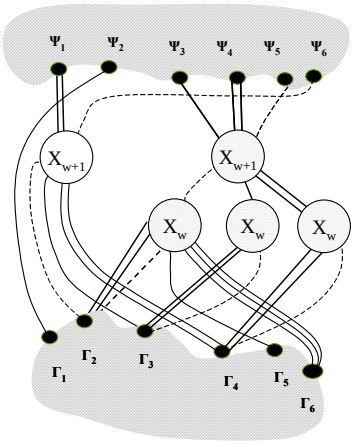


Fig. 8. Step 7

After these seven steps we have successfully swapped two adjacent variables in a TED without changing the subgraphs above or below the swapping variables. The correctness of the algorithm relies on the proof of theorem I. The proof that the algorithm terminates is given by the fact that a finite multivariate polynomial has a finite number of monomials, therefore there are a finite number of paths and the expansion produced in the first five steps is finite.

B. Complexity of the local swapping

The running time of the algorithm will be related to the maximum number of expansion ξ that can take place, and this can be calculated in general by:

$$\xi = \sum_{i=1}^{\#Nodes(x_{w+1})} (\#k_c Edges\_to\_\Gamma \times \#k_p Edges\_to\_x_w)_{Node(x_{w+1})_i}$$

using the inequality $ac + bd < (a+b)(c+d)$ for positive integer numbers, we conclude that the running time of the algorithm is *O(mn)*, with *m* equal to the total number of edges connecting x_{w+1} to Γ, and *n* equal to the total number of edges connecting x_{w+1} to x_w.

Observation: In all the examples we have tested so far, reducing the number of nodes in a TED did not increase the number of edges.

C. Dynamic ordering

Local swapping can be achieved with any of the known dynamic ordering algorithms (proposed for BDDs, OBDDs, etc. [3] [4] [5] [6] [7] [8] [9] [10]) that are based on a greedy property. More specifically, we can perform the sifting

algorithm and window permutation; by bubbling up, bubbling down and backtracking the best position can be found for a given variable.

D. Static ordering

We also propose a fast way of variable allocation that can minimize the search space. This ordering is heuristic and is based on the following observations for TED graphs.

Observation 1: Variables that appear in most terms of the monomial with the same exponent in most terms should be placed at the top, as they will produce only one node and least edges.

Observation 2: Variables that appear in most terms of the monomials and have multiple number of exponents should be placed right after any homogeneous variable identified in the first observation.

Observation 3: In the case of a TED with a single output the two first observations are followed. In the case of multiple outputs we change *top* for *bottom* and *after* for *before* in the two first observations; the latter is done to maximize sharing of TED nodes. (see Fig. 9)

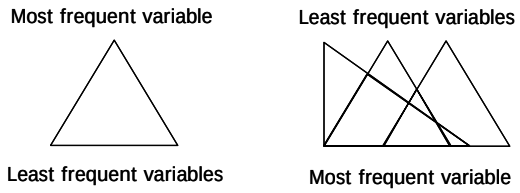


Fig. 9. Single and Multiple output TEDs

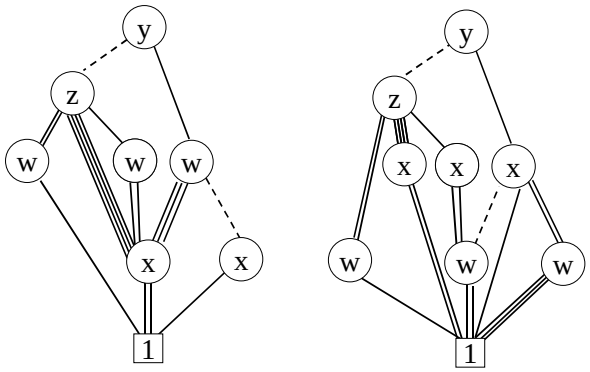
Observation 4: Variables that never appear in the same monomial are treat as outputs of a multiple output TED. The idea behind this observation is that treating the variables as outputs of a multiple output TED allows to fit each variable into the first observation. And all these unrelated variables will be joined by the additive edge to produce the single output TED that we were building in the first place.

An easy way to identify these cases for multivariate polynomials, given in the expanded form, is by constructing a matrix whose columns represent monomials and rows represent the variables. In each element of the matrix *(i,j)* we put the respective exponent *k* with variable x_i that appears in the monomial *j*.

For example the following multivariate polynomial $f(x, y, w, z) = x^2w^3y + xy + z^4x^2 + wz^2 + x^2w^2z + w^2y$ will produce the following matrix:

	x^2w^3y	xy	z^4x^2	wz^2	x^2w^2z	w^2y
x	2	1	2	0	2	0
y	1	1	0	0	0	1
w	3	0	0	1	2	2
z	0	0	4	2	1	0

From the matrix the initial variable order is $y < z < w < x$. We have chosen *y* and *z* at the top because of the forth observation, *x* is placed at the bottom because it has the same number of calls as *w* but it has more terms with the same exponent. Now the only possible further search in variable optimization would be swapping variables *w* and *x*, and this means that we have effectively reduce our space search from 4! = 24 to 2. (see Fig. 10)



a) $y < z < w < x$ b) $y < z < x < w$

Fig. 10. Reduced search space for $f(x, w, y, z)$.

Conclusions

From the analysis presented in this paper we can conclude the following:

1. Swapping two adjacent variables doesn't change the graphs above and below the variables and furthermore it does not change the edges and weights connecting the graph above.
2. We can apply the algorithms known from dynamic ordering for BDDs that are based on the greedy property of local optimality.
3. Reducing the number of nodes and reducing the number of edges in a TED are the same objective.

Recently TED was shown to outperform Mathematica in both time and space when checking for equality of large polynomials [11]. This result is expected to be improved using reordering techniques described on this paper.

References

[1] M. Ciesielski, P. Kalla, Z. Zheng, and B. Rouzyere, "Taylor Expansion Diagrams: A Canonical Representation for High-Level Design Verification", Proc. Design Automation and Test in Europe, DATE-02, 2002, pp. 285-289.

[2] Eric W. Weisstein, "Partial Derivative." From MathWorld A Wolfram Web Resource. http://mathworld.wolfram.com/PartialDerivative.html

[3] R. Rudell, "Dynamic variable ordering for ordered binary decision diagrams", In Proceedings of the International Conference on Computer-Aided Design, Santa Clara, CA, November 1993, pp 42-47.

[4] W. Günther, R. Drechsler, "BDD minimization by linear transformation", Great Lakes Symposium on VLSI '98 , Lafayette (Louisiana, USA), February 1998, pp 325-330.

[5] Ch. Meinel, F. Somenzi, T. Theobald, "Linear Sifting of Decision Diagrams", Proc. 34th IEEE/ACM DAC, Anaheim (USA), IEEE Computer Society Press, 1997, pp 202-207.

[6] Ch. Meinel, A. Slobodová, "Speeding up Variable Ordering of OBDDs", Proc. of ICCD 97, Austin (Texas, USA), 1997, pp. 338-343.

[7] S. Panda, F. Somenzi, and B. F. Plessier, "Symmetry detection and dynamic variable ordering of decision diagrams", In Proceedings of the International Conference on Computer-Aided Design, San Jose, CA, November 1994, pp 628-631.

[8] S. Panda and F. Somenzi, "Who are the variables in your neighborhood", In Proceedings of the International Conference on Computer-Aided Design, San Jose, CA, November 1995, pp 74-77.

[9] Moller, D., Molitor, P., Drechsler, R., "Symmetry Based Variable Ordering for ROBDDs", IFIP Workshop on Logic and Architecture Synthesis, 1994, pp. 47-53.

[10] R. Ebendt, "Reducing the Number of Variable Movements in Exact BDD Minimization", In Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS'03), Bangkok, 2003.

[11] Q. Rian, S. Askar, M. Ciesielski, "Taylor Expansion Diagram Versus Mathematica: Evaluation by Simulation", Department of Electical Engineering at University of Massachusetts, TR-04-CSE-15, 2004.

.